Supervised Learning

John Joseph Valletta

University of Exeter, Penryn Campus, UK

April 2019

Overview

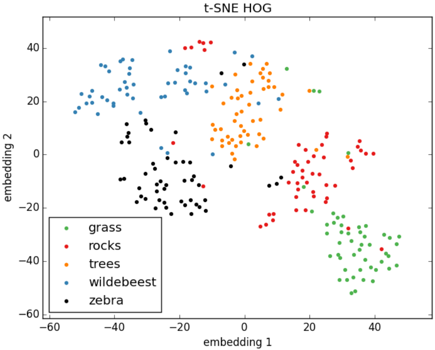
- What is supervised learning?
- Cross-validation
- k-nearest neighbour (kNN)
- Decision trees
- Random forests
- Support vector machines

What's the problem?

What's the problem?

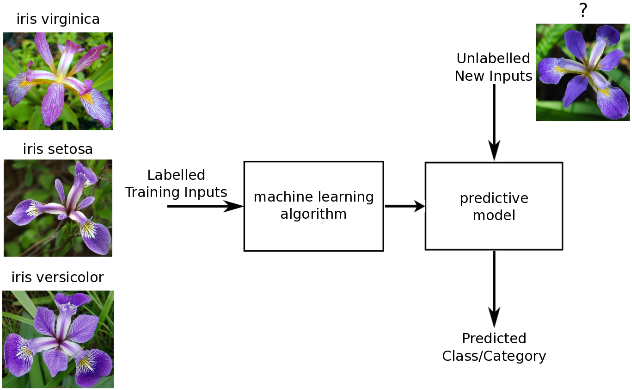
https://www.livescience.com/23310-serengeti.html

Counting wildebeest



What is supervised learning?

Supervised learning methods determine the mapping (**predictive model**) between a set of features and a continuous outcome (**regression**), or a categorical variable (**classification**)

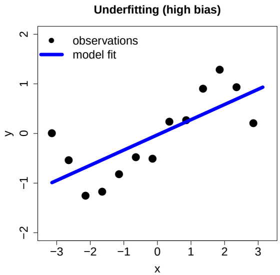


Bias-variance tradeoff

- Machine learning algorithms are very flexible in order to deal with complex data
- So how *well* should we fit to the training data to get good generalisation?
- Driving the training error to zero is *not* a good idea

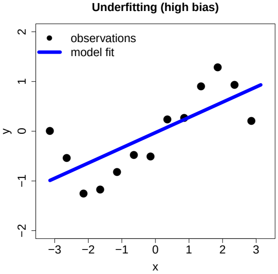
Bias-variance tradeoff

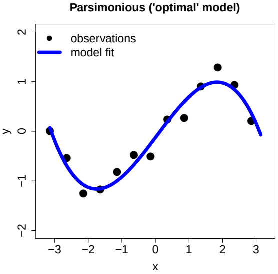
- Machine learning algorithms are very flexible in order to deal with complex data
- So how *well* should we fit to the training data to get good generalisation?
- Driving the training error to zero is *not* a good idea



Bias-variance tradeoff

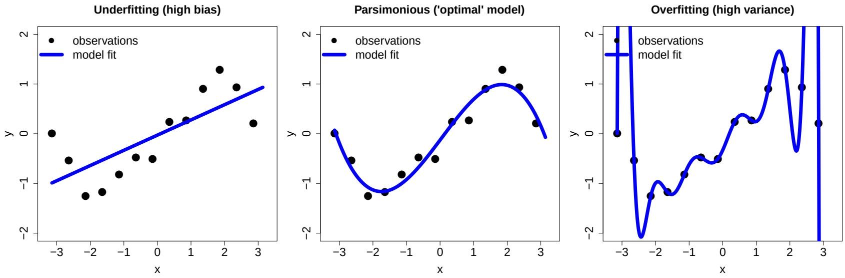
- Machine learning algorithms are very flexible in order to deal with complex data
- So how *well* should we fit to the training data to get good generalisation?
- Driving the training error to zero is *not* a good idea





Bias-variance tradeoff

- Machine learning algorithms are very flexible in order to deal with complex data
- So how *well* should we fit to the training data to get good generalisation?
- Driving the training error to zero is *not* a good idea

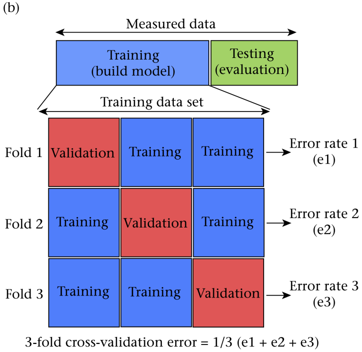


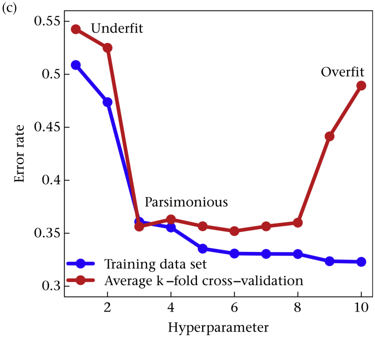
Cross-validation

- The flexibility of ML models is constrained by **tuning** their **hyperparameters**
- k-**fold cross-validation** allow us to find optimal hyperparameters

Cross-validation

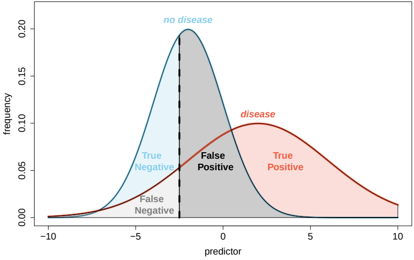
- The flexibility of ML models is constrained by **tuning** their **hyperparameters**
- k-**fold cross-validation** allow us to find optimal hyperparameters

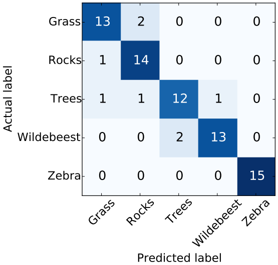




Predictive performance measures

- To compare models during cross-validation **predictive performance measures** are computed
- Several metrics exist, some of the more popular ones are:
  - **Regression**: root mean squared error (RMSE), R-squared
  - **Classification**: area uder the ROC curve, confusion matrix





k-nearest neighbour (kNN)

1. Calculate distance between test point and every training data point
2. Find the k training points closest to test point
3. Assign test point the majority vote of their class label

k-nearest neighbour (kNN)

1. Calculate distance between test point and every training data point
2. Find the k training points closest to test point
3. Assign test point the majority vote of their class label

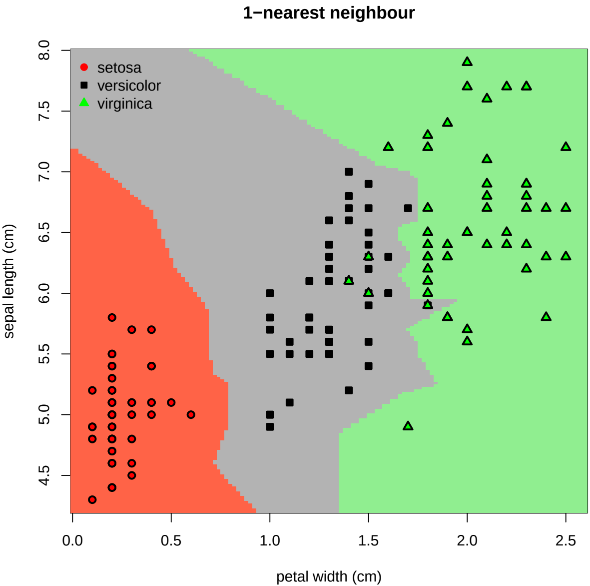
k-nearest neighbour (kNN)

1. Calculate distance between test point and every training data point
2. Find the k training points closest to test point
3. Assign test point the majority vote of their class label

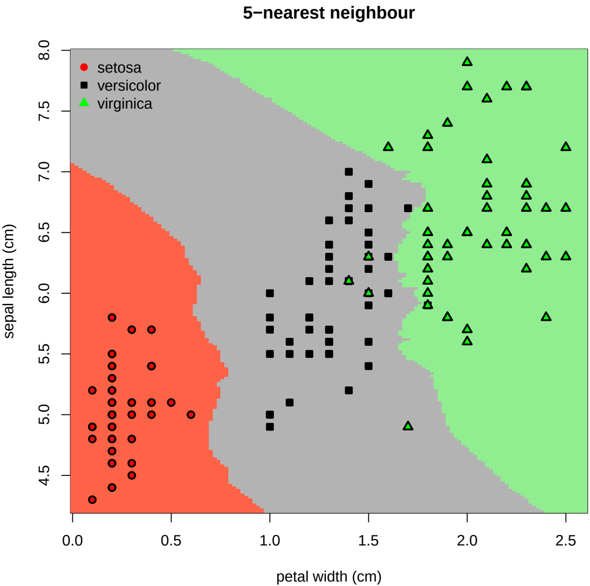
k-nearest neighbour (kNN)

1. Calculate distance between test point and every training data point
2. Find the k training points closest to test point
3. Assign test point the majority vote of their class label

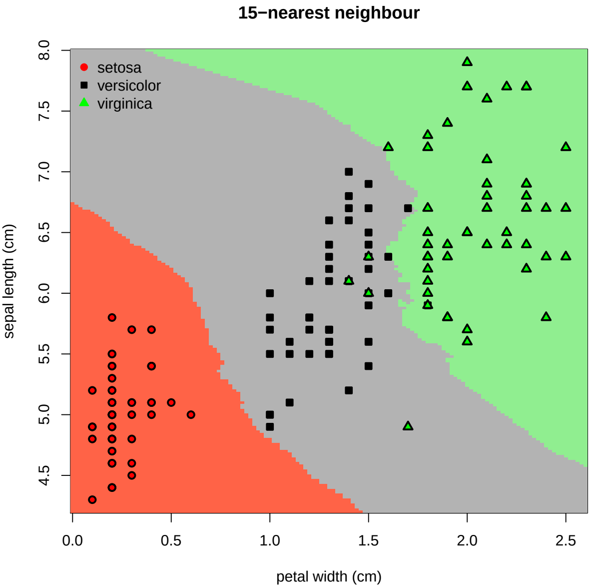
k-nearest neighbour



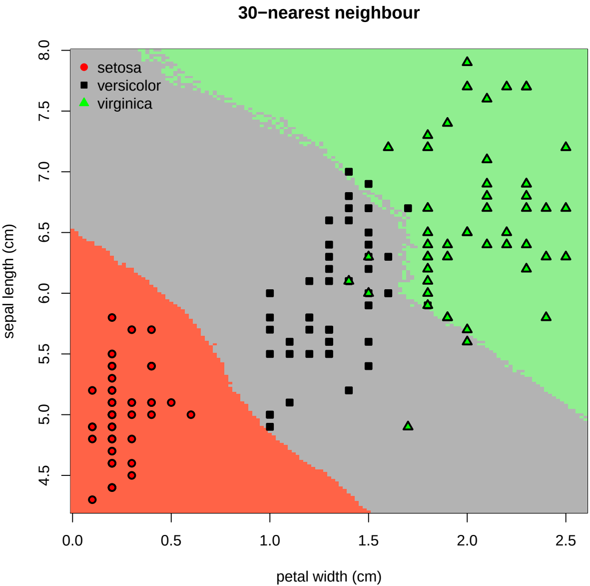
k-nearest neighbour



k-nearest neighbour



k-nearest neighbour



k-nearest neighbour

Pros

- Simple and intuitive
- Works for multi-class problems
- Non-linear decision boundaries
- k easily tuned by cross-validation

Cons

- Can be computationally expensive, as for every test point, distance to *every* training data point needs to be computed
- Takes up a lot of storage as *all* training points need to be retained

Decision trees

1. Find the yes/no rule that best splits the data with respect to *one* of the features

2. The best split is the one that produces the most homogeneous groups; found by maximising information gain/lowering entropy.

3. Repeat steps 1 to 2 until all data are correctly classified or some stopping rule reached.

Decision trees

1. Find the yes/no rule that best splits the data with respect to *one* of the features

2. The best split is the one that produces the most homogeneous groups; found by maximising information gain/lowering entropy.

3. Repeat steps 1 to 2 until all data are correctly classified or some stopping rule reached.

Decision trees

1. Find the yes/no rule that best splits the data with respect to *one* of the features

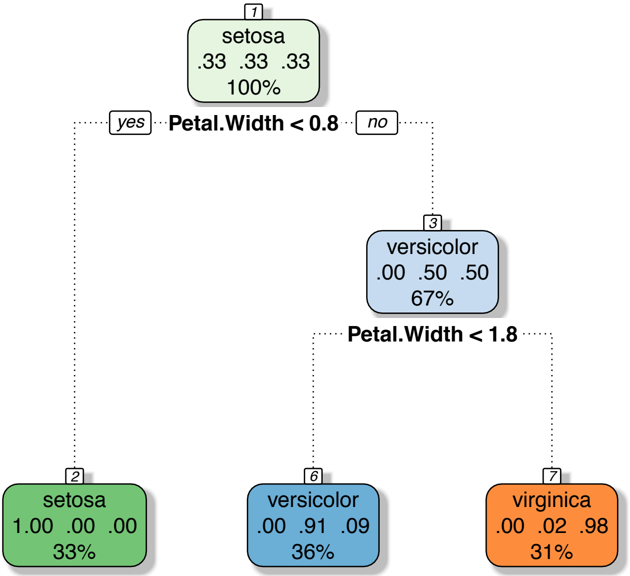
2. The best split is the one that produces the most homogeneous groups; found by maximising information gain/lowering entropy.

3. Repeat steps 1 to 2 until all data are correctly classified or some stopping rule reached.

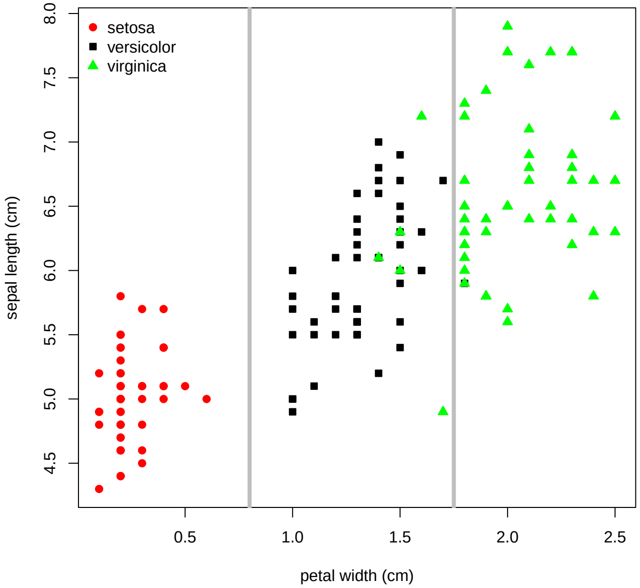
Decision trees

1. Find the yes/no rule that best splits the data with respect to *one* of the features
2. The best split is the one that produces the most homogeneous groups; found by maximising information gain/lowering entropy.
3. Repeat steps 1 to 2 until all data are correctly classified or some stopping rule reached.

Decision trees



Decision boundaries



Decision trees

Pros

- Model is very easy to explain to non-experts and can be directly used to generate rules
- Computationaly inexpensive to train, evaluate and store
- Handle both categorical and continuous data
- Robust to outliers

Cons

- Can easily overfit the data
- Predictive accuracy can be poor
- Linear decision boundaries
- Small changes to training data may lead to a completely different tree

Random forests

- Decision trees are intuitive but suffer from overfitting which significantly affect their predictive accuracy
- **Pruning**, to "trim" the tree back, help reduce this overfit
- **Ensemble** methods such as Random Forests are a better alternative
- **Rationale**: Instead of one tree, grow a *forest*, where every bushy tree (no pruning) is a bit different, then average predictions over all trees

Random forests

- Decision trees are intuitive but suffer from overfitting which significantly affect their predictive accuracy
- **Pruning**, to "trim" the tree back, help reduce this overfit
- **Ensemble** methods such as Random Forests are a better alternative
- **Rationale**: Instead of one tree, grow a *forest*, where every bushy tree (no pruning) is a bit different, then average predictions over all trees

Random forests

- Decision trees are intuitive but suffer from overfitting which significantly affect their predictive accuracy
- **Pruning**, to "trim" the tree back, help reduce this overfit
- **Ensemble** methods such as Random Forests are a better alternative
- **Rationale**: Instead of one tree, grow a *forest*, where every bushy tree (no pruning) is a bit different, then average predictions over all trees

Random forests

- Decision trees are intuitive but suffer from overfitting which significantly affect their predictive accuracy
- **Pruning**, to "trim" the tree back, help reduce this overfit
- **Ensemble** methods such as Random Forests are a better alternative
- **Rationale**: Instead of one tree, grow a *forest*, where every bushy tree (no pruning) is a bit different, then average predictions over all trees

Random forests

- Decision trees are intuitive but suffer from overfitting which significantly affect their predictive accuracy
- **Pruning**, to "trim" the tree back, help reduce this overfit
- **Ensemble** methods such as Random Forests are a better alternative
- **Rationale**: Instead of one tree, grow a *forest*, where every bushy tree (no pruning) is a bit different, then average predictions over all trees

$\Longrightarrow$

Random forests

1. Grow T decorrelated trees (no pruning)
2. Forest randomness is induced by:
3. Average predictions from all T trees.

Random forests

1. Grow T decorrelated trees (no pruning)

2. Forest randomness is induced by:

3. Average predictions from all T trees.

Random forests

1. Grow T decorrelated trees (no pruning)

2. Forest randomness is induced by:
   - Bagging (**B**ootstrap **AGG**regat**ING**), each tree is trained on a subset of the data randomly sampled with replacement
   - Considering only a subset of predictors as candidates for each split

3. Average predictions from all T trees.

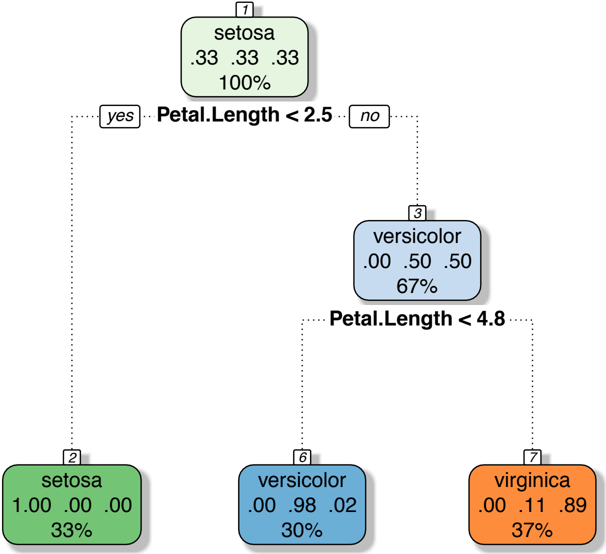
Random forests

1. Grow T decorrelated trees (no pruning)

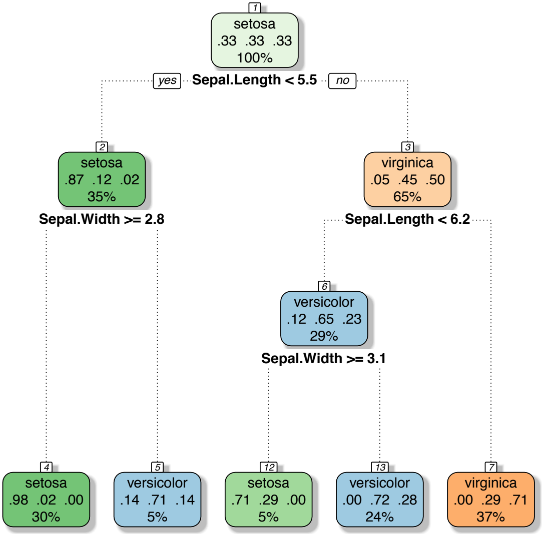
2. Forest randomness is induced by:
   - Bagging (**B**ootstrap **AGG**regat**ING**), each tree is trained on a subset of the data randomly sampled with replacement
   - Considering only a subset of predictors as candidates for each split

3. Average predictions from all T trees.

De-correlated trees



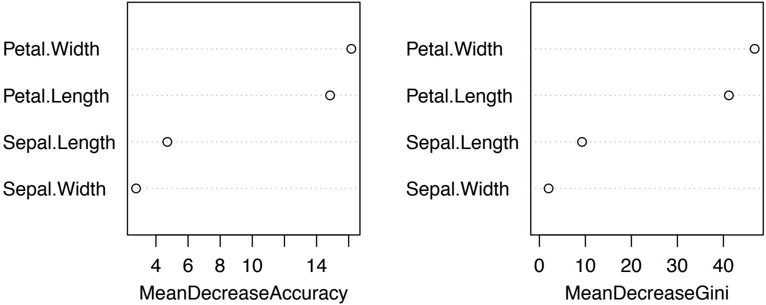
De-correlated trees



Variable importance

- Cannot visualise decision boundaries (loss of interpretability)

- However, variable importance helps us perform feature selection



Random forests

Pros

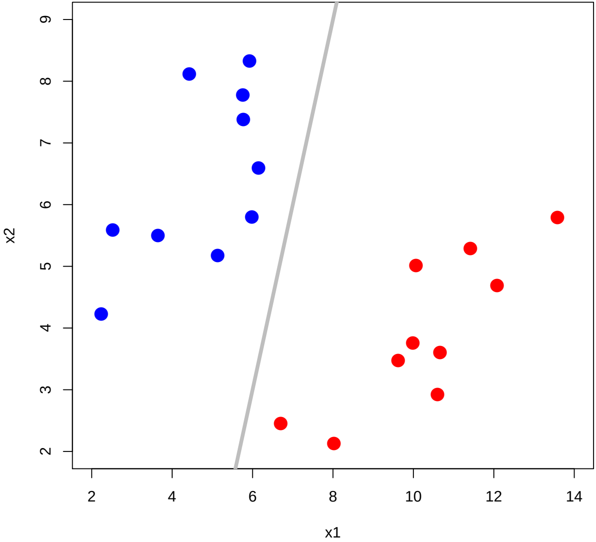
- State-of-the-art predictive accuracy
- Can handle thousands of both categorical and continuous predictors without variable deletion
- Robust to outliers
- Estimates the importance of every predictor
- Out-of-bag error (unbiased estimate of test error for every tree built)
- Can cope with unbalanced datasets by setting class weights
- Trivially parallelisable

Cons

- Harder to interpret then plain decision trees

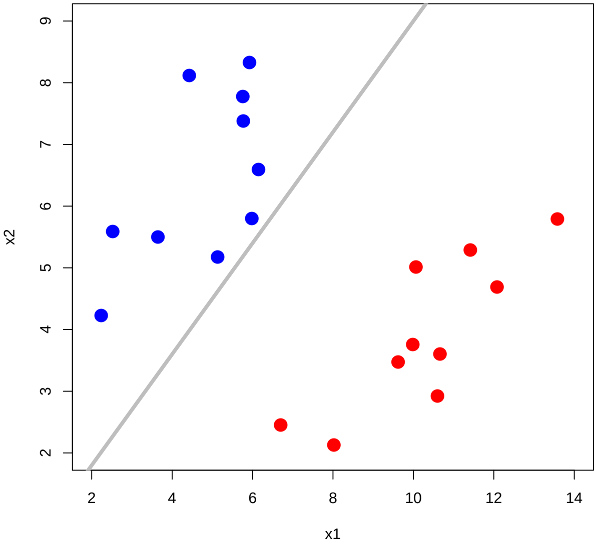
Support vector machines (SVMs)

Which is the best separating line?



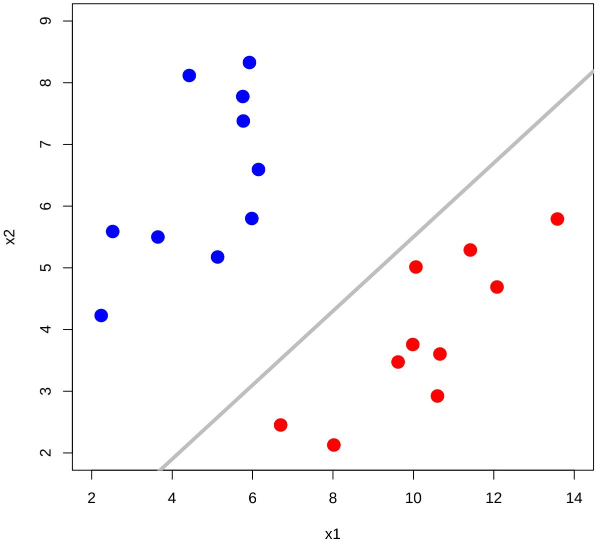
Support vector machines (SVMs)

Which is the best separating line?

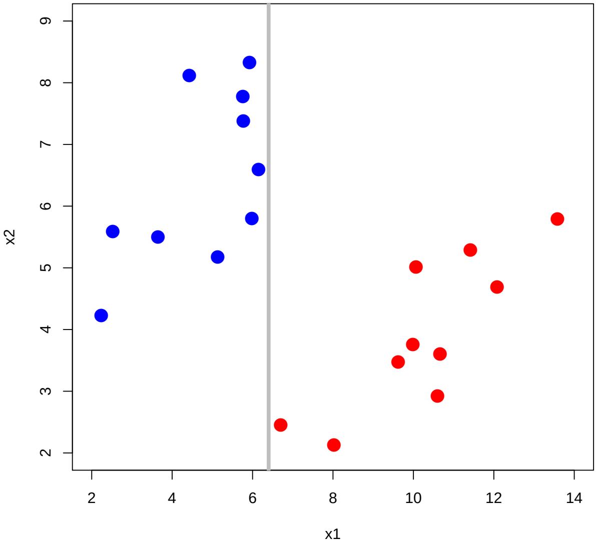


Support vector machines (SVMs)

Which is the best separating line?

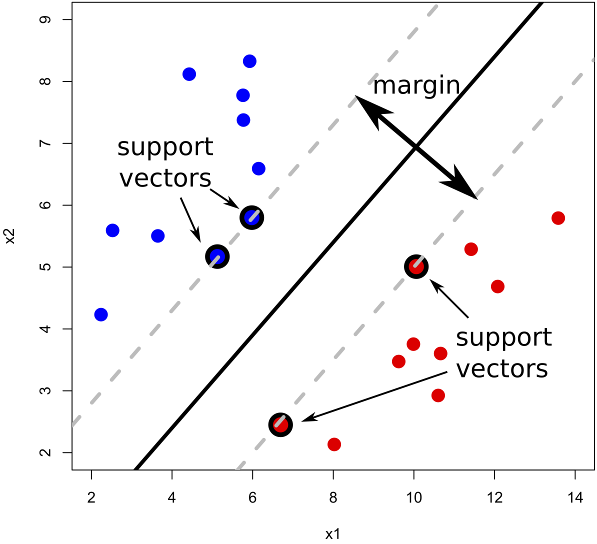


Which is the best separating line?

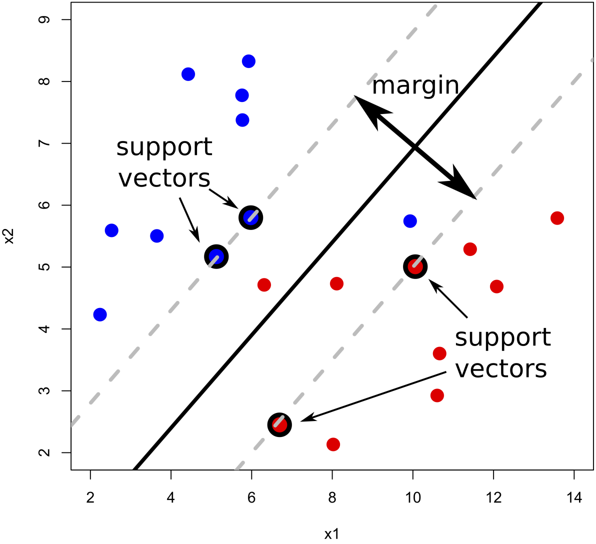


Maximal margin classifier

Rationale: Maximise the *margin*

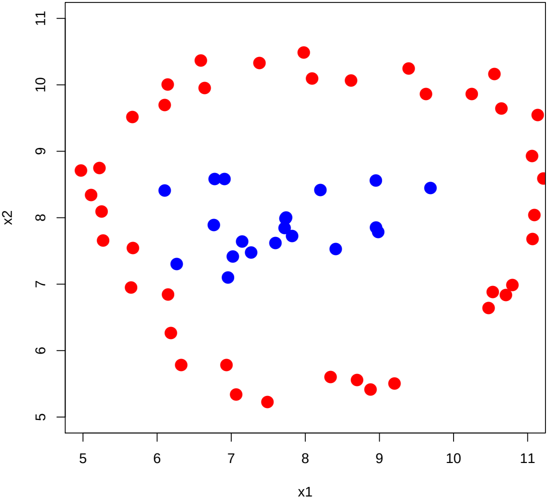


Rationale: Use a *soft* margin



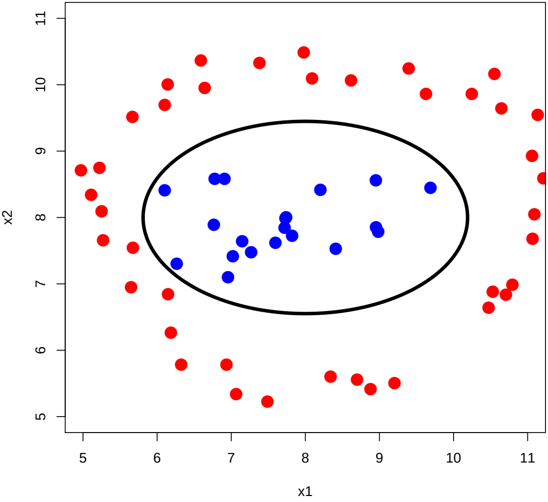
Support vector machines

Real-life data is complex and often we cannot find a separating hyperplane

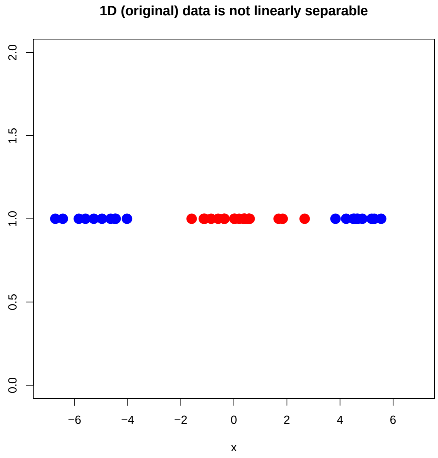


Support vector machines

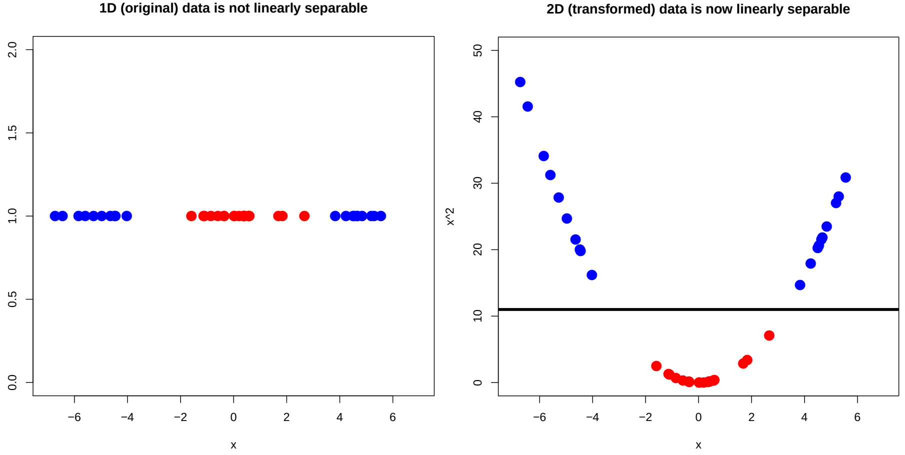
Rationale: Map data to a higher dimensional space where classes are linearly separable



Support vector machines: 1D to 2D



Support vector machines: 1D to 2D



Support vector machines - the kernel trick

- So our solution is to blow up the dimensions?
- But what about the “curse of dimensionality”?
- Very computationally expensive to work in high dimensions

Support vector machines - the kernel trick

- So our solution is to blow up the dimensions?
- But what about the "curse of dimensionality"?
- Very computationally expensive to work in high dimensions

- **Kernel trick** to the rescue!
- Work in an *implict* feature space
- Data is never explicitly computed in higher dimensions
- Think about kernels as generalised distance measures

p.s Kernel methods are mathematically intricate and beyond the scope of this introductory workshop

Support vector machines

1. Choose a kernel

2. Run optimiser to find the maximum margin separating hyperplane

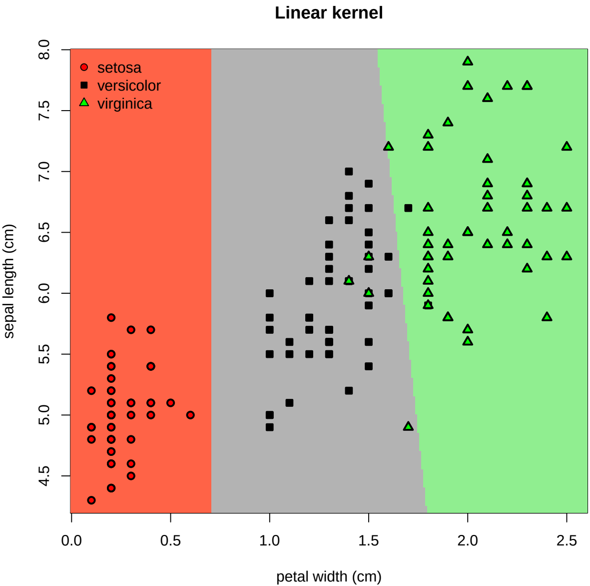
Support vector machines

1. Choose a kernel

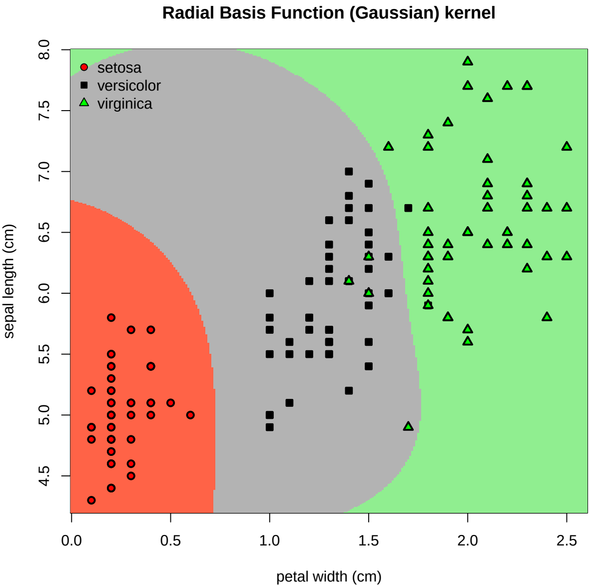
2. Run optimiser to find the maximum margin separating hyperplane

SVMs are inherently binary classifiers. The most common ways to deal with multi-class problems is by building several **one-versus-all** *or* **one-versus-one** classifiers.

Support vector machines



Support vector machines



Support vector machines

Pros

- State-of-the-art predictive accuracy
- Low storage requirements (only support vectors to store)
- A vast array of kernels are available that are flexible enough to cater for any type of data
- Global optimum guaranteed

Cons

- Model is hard to interpret
- Feature space cannot be visualised